Performance Modelling — Lecture 16: Model Validation and Verification

Jane Hillston
School of Informatics
The University of Edinburgh
Scotland

16th March 2017

Introduction

Whatever modelling paradigm or solution technique is being used, the performance measures extracted from a model will only have some bearing on the real system represented if the model is a good representation of the system.

Introduction

Whatever modelling paradigm or solution technique is being used, the performance measures extracted from a model will only have some bearing on the real system represented if the model is a good representation of the system.

What constitutes a good model is subjective.

Introduction

Whatever modelling paradigm or solution technique is being used, the performance measures extracted from a model will only have some bearing on the real system represented if the model is a good representation of the system.

What constitutes a good model is subjective.

From a performance modelling point of view our criteria for judging the goodness of models will be based on how accurately measures extracted from the model correspond to the measures which would be obtained from the represented system.

Two faces of abstraction

By its nature a model is more abstract than the system it represents (even a simulation model).

Two faces of abstraction

By its nature a model is more abstract than the system it represents (even a simulation model).

Viewed in one way, abstraction, and assumptions we make to achieve it, eliminate unnecessary detail and allow us to focus on the elements within the system which are important;

Two faces of abstraction

By its nature a model is more abstract than the system it represents (even a simulation model).

Viewed in one way, abstraction, and assumptions we make to achieve it, eliminate unnecessary detail and allow us to focus on the elements within the system which are important;

Viewed in another way, abstraction introduces inaccuracy, because it means that the model is not entirely faithful to the system.

Two faces of abstraction

By its nature a model is more abstract than the system it represents (even a simulation model).

Viewed in one way, abstraction, and assumptions we make to achieve it, eliminate unnecessary detail and allow us to focus on the elements within the system which are important;

Viewed in another way, abstraction introduces inaccuracy, because it means that the model is not entirely faithful to the system.

Some degree of inaccuracy may be necessary, desirable even, to make the model solution tractable and/or efficient.

Modelling assumptions

- Inevitably some assumptions must be made about the system in order to construct the model.

Modelling assumptions

- Inevitably some assumptions must be made about the system in order to construct the model.
- Having made such assumptions we must expect to put some effort into answering questions about the goodness of our model.

Modelling assumptions

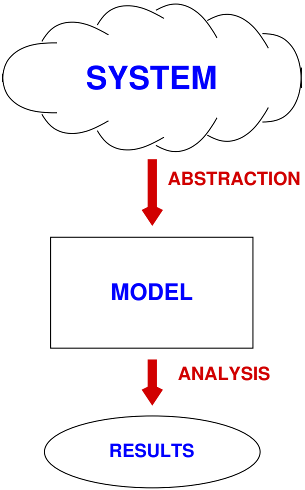
- Inevitably some assumptions must be made about the system in order to construct the model.
- Having made such assumptions we must expect to put some effort into answering questions about the goodness of our model.
- There are two steps to judging how good a model is with respect to the system.

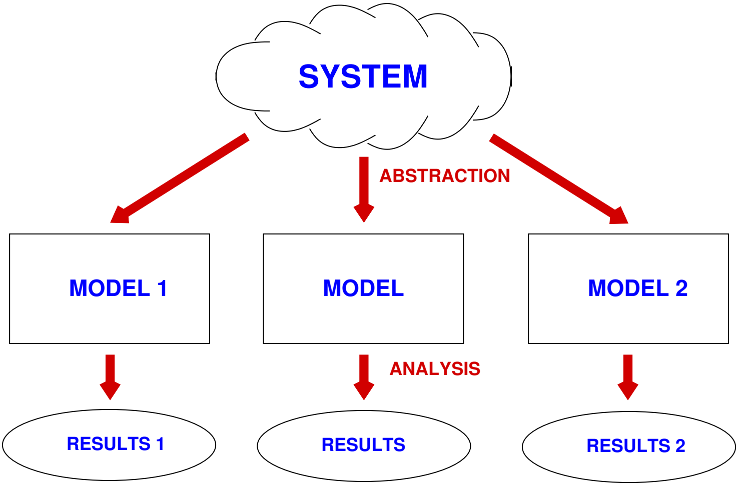
Modelling assumptions

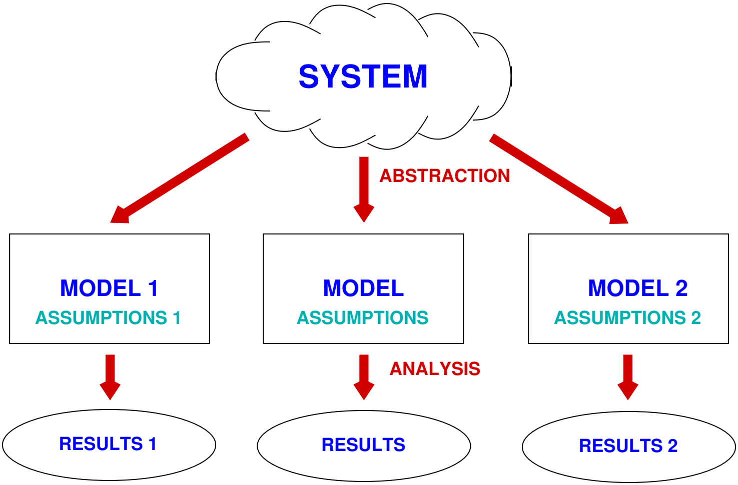
- Inevitably some assumptions must be made about the system in order to construct the model.
- Having made such assumptions we must expect to put some effort into answering questions about the goodness of our model.
- There are two steps to judging how good a model is with respect to the system.
  1. We must ascertain whether the model implements the assumptions correctly (model verification); and

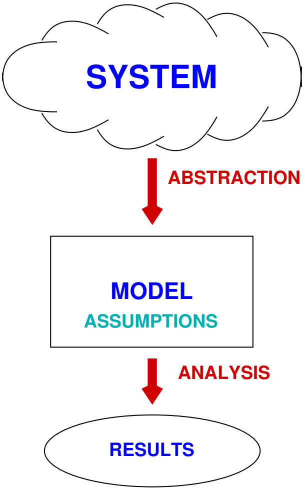
Modelling assumptions

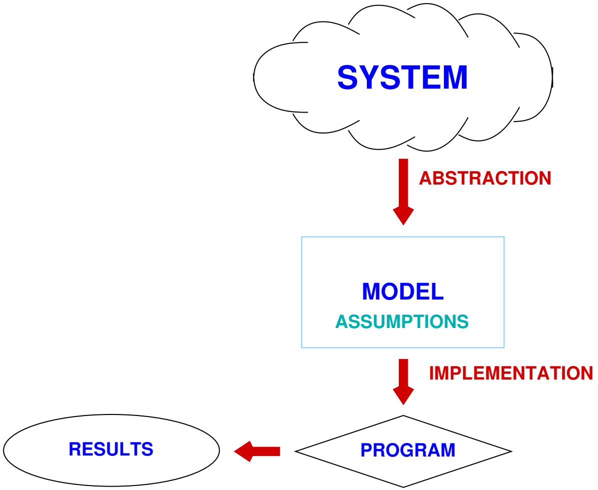
- Inevitably some assumptions must be made about the system in order to construct the model.
- Having made such assumptions we must expect to put some effort into answering questions about the goodness of our model.
- There are two steps to judging how good a model is with respect to the system.
  1. We must ascertain whether the model implements the assumptions correctly (model verification); and
  2. whether the assumptions which have been made are reasonable with respect to the real system (model validation).

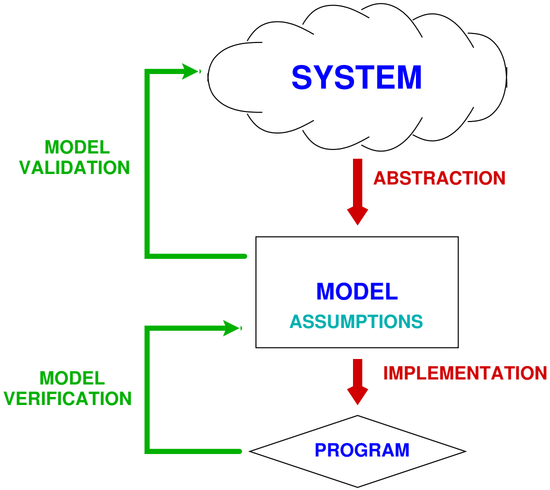

SYSTEM
ABSTRACTION
MODEL
ANALYSIS
RESULTS


SYSTEM
ABSTRACTION
MODEL 1
MODEL
MODEL 2
ANALYSIS
RESULTS 1
RESULTS
RESULTS 2


SYSTEM
ABSTRACTION
MODEL 1
ASSUMPTIONS 1
MODEL
ASSUMPTIONS
MODEL 2
ASSUMPTIONS 2
ANALYSIS
RESULTS 1
RESULTS
RESULTS 2


SYSTEM
ABSTRACTION
MODEL
ASSUMPTIONS
ANALYSIS
RESULTS


SYSTEM
ABSTRACTION
MODEL
ASSUMPTIONS
IMPLEMENTATION
PROGRAM
RESULTS


SYSTEM
ABSTRACTION
MODEL
ASSUMPTIONS
IMPLEMENTATION
PROGRAM
MODEL
VALIDATION
MODEL
VERIFICATION

Model verification

In the Markov process model of the PC-LAN we carried out a walk-through of the simplified model with just two nodes in the network to check the characterisation of state which we were using.

This was model verification.

Model verification

In the Markov process model of the PC-LAN we carried out a walk-through of the simplified model with just two nodes in the network to check the characterisation of state which we were using.

This was model verification.

The level of abstraction which we had first chosen did not allow us to represent the behaviour of the system and make the assumptions about memoryless behaviour.

Model verification

In the Markov process model of the PC-LAN we carried out a walk-through of the simplified model with just two nodes in the network to check the characterisation of state which we were using.

This was model verification.

The level of abstraction which we had first chosen did not allow us to represent the behaviour of the system and make the assumptions about memoryless behaviour.

The walk-through allowed us to detect that we needed to distinguish states more finely, leading to a modification of the model.

Verification with validation

- It is important to remember that

Verification with validation

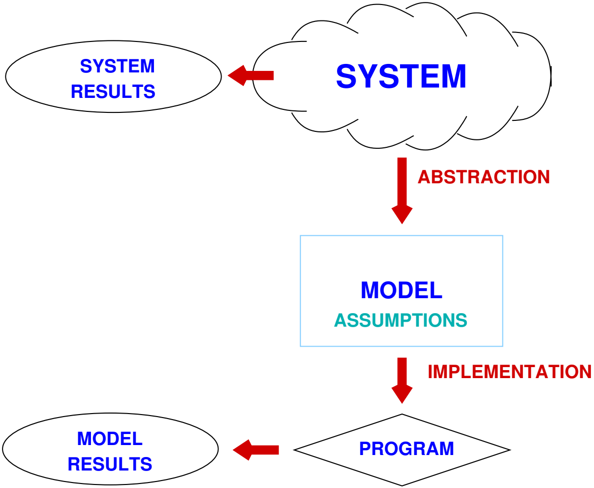
- It is important to remember that
  - validation does not imply verification, and

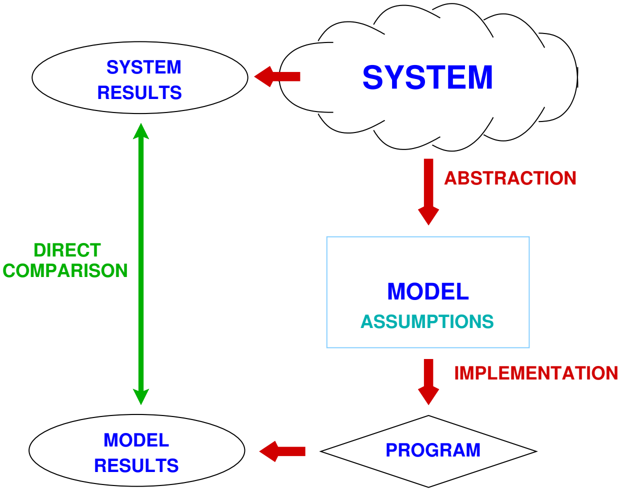
Verification with validation

- It is important to remember that
  - validation does not imply verification, and
  - verification does not imply validation.

Verification with validation

- It is important to remember that
  - validation does not imply verification, and
  - verification does not imply validation.
- In practice, validation is often blended with verification, especially when measurement data is available for the system being modelled.


SYSTEM RESULTS
SYSTEM
ABSTRACTION
MODEL
ASSUMPTIONS
IMPLEMENTATION
MODEL RESULTS
PROGRAM


SYSTEM
RESULTS
SYSTEM
ABSTRACTION
DIRECT
COMPARISON
MODEL
ASSUMPTIONS
IMPLEMENTATION
MODEL
RESULTS
PROGRAM

Model verification

- Verification is like debugging—it is intended to ensure that the model does what it is intended to do.
- Models, especially simulation models, are often large computer programs.
- Techniques that can help develop, debug or maintain large computer programs are also useful for models.
- Many authors advocate modularity and top-down design.

Antibugging

- Antibugging consists of including additional checks and outputs in a model that may be used to capture bugs if they exist.
- These are features of the model which do not have a role in representing the system, or even necessarily in calculating performance measures. Their only role is to check the behaviour of the model.

Antibugging in simulation

- A common form of antibugging is to maintain counters within a simulation model which keep track of the number of entities which are generated and terminated during the evolution of the model.
- In a communication network simulation we might keep track of the number of messages entering the network, the number successfully delivered and the number lost.
- Then, at any time, the number of messages which have entered should be equal to the number which have been delivered plus the number that have been lost plus the number which are currently in transit.

Antibugging in Markovian models

- In Markovian models it is not possible to maintain counters in this explicit way as it would break the irreducibility assumption about the state space.
- However, it is still sometimes possible to include antibugging structures within a Markovian model.
- For example, an extra place could be included in a GSPN model into which a token was inserted for each entity entering the model and a token was removed for each entity that left.
- Care should be exercised in applying such techniques to ensure that you are not creating state space explosion.

Structured walk-through/one-step analysis

- Explaining the model to someone else can make the modeller focus on different aspects of the model and therefore discover problems with its current implementation.
- The listeners do not need to be experts in the system — the developer may become aware of bugs simply by studying the model carefully and trying to explain how it works.
- Preparing documentation for a model can have a similar effect by making the modeller look at the model from a different perspective.

Single-stepping through models

- Alternatively the model developer should try to carry out the same sort of step-by-step analysis of the model to convince himself or herself that it behaves correctly.
  - For a GSPN model this would be playing the token game;
  - In a queueing network, stepping through the possible customer transitions.
- Some modelling packages provide support for doing this. The PEPA Eclipse Plug-in has a single-step navigator.

Simplified models

- It is sometimes possible to reduce the model to its minimal possible behaviour.
- We used this for the “multiprocessor” example with only one processor, to make sure that the interaction between the processor and the common memory was correct.
- Similarly when we considered the PC-LAN with just two PCs.
  - In a closed queueing network model we might consider the model with only a single customer.
  - In a process-based simulation model we might only instantiate one entity of each type.
- Since one-step analysis can be extremely time consuming it is often applied to a simplified model.

Use of simplified models

Of course, a model that works for simple cases is not guaranteed to work for more complex cases.

Use of simplified models

Of course, a model that works for simple cases is not guaranteed to work for more complex cases.

On the other hand, a model which does not work for simple cases will certainly not work for more complex ones.

Deterministic models

Simulation only

- For simulation models the presence of random variables can make it hard for the modeller to reason about the behaviour of a model and check that it is as expected or required.
- Replacing random variables which govern delays or scheduling with deterministic values may help the modeller to see whether the model is behaving correctly.
- Only when we are satisfied that the behavioural representation of the entities is indeed correct should we introduce random variables to represent inter-event times using continuous time distributions.

Tracing

Simulation only

- A trace is a sequence of events and the times at which they occur.
- Traces can be useful in isolating incorrect behaviour in a model, although in general other techniques will be used to identify the presence of a bug in the first place.
- Since tracing causes considerable additional processing overhead it should be used sparingly in all except the simplest models.
- It should always be switched off once the modeller is confident that the model is behaving correctly.

Animation

Simulation only

- From a verification perspective, animation is similar to tracing but provides the information about the internal behaviour of the model in a graphical form.
- Some modelling packages with graphical interfaces provide a dynamic display of model behaviour whilst the model is executing.
- In some commercial tools the display will represent high level information about the current value of the performance measures of interest shown as dials or meters which change as the values change.

Visualisation

- Other animation views are more low-level, showing events on a graphical representation of the system, possibly explicitly showing the interactions between entities.
- This low level view is particularly suited to verification.
- Animation can take the form of automated one-step analysis, if the animation facilities allow the view of the model to advance one event at a time.

Visualisation and Petri nets

- Graphical stochastic Petri net and queueing network tools often provide a animated form of one-step analysis in which tokens or customers can be seen moving around the network.
- Handling the display as well as the evolution of the model slows down the simulation considerably.
- As with tracing, animation is most useful for isolating an error once its presence has been established.

Seed independence

Simulation only

- The seeds used for random number generation in a simulation model should not significantly affect the model outcomes, although there will be variation in sample points as seeds vary.
- If a model produces widely varying results for different seed values it indicates that there is something wrong within the model.
- Seed independence can be verified by running the simulation with different seed values, something which is probably necessary in any case.

Continuity testing

- At an abstract level all systems and models can be thought of as generating a function from input values to output values,

$$\mathcal{F}(\mathbf{I}) = \mathbf{O}$$

Continuity testing

- At an abstract level all systems and models can be thought of as generating a function from input values to output values,

$$\mathcal{F}(\mathbf{I}) = \mathbf{O}$$

- and in most cases we expect that function to be continuous

$$\mathcal{F}(\mathbf{I} + \delta) = \mathbf{O} + \delta.$$

Continuity testing

- At an abstract level all systems and models can be thought of as generating a function from input values to output values,

$$\mathcal{F}(\mathbf{I}) = \mathbf{O}$$

- and in most cases we expect that function to be continuous

$$\mathcal{F}(\mathbf{I} + \delta) = \mathbf{O} + \delta.$$

- In other words, in most cases, we do not anticipate that a slight change in an input value will result in very large changes in the corresponding output value.

Continuity testing and models

- Continuity testing consists of running a simulation model, or solving a Markovian model, several times for slightly different values of input parameters.
- For any one parameter, a slight change in input should generally produce only a slight change in the output.
- Any sudden changes in the output are taken to be an indication of a possible error which should be investigated unless this is known behaviour of the system.

Degeneracy testing

- Degenerate cases for a model are those values of input parameters which are at the extremes of the model's intended range of representation.
- Degeneracy testing consists of checking that the model works for the extreme values of system and workload (input) parameters.
- Although extreme cases may not represent typical cases, degeneracy testing can help the modeller to find bugs that would not otherwise have been discovered.

Consistency testing

- For most models and systems it is reasonable to assume that similarly loaded systems will exhibit similar characteristics, even if the arrangement of the workload varies.
- Consistency tests are used to check that a model produces similar results for input parameter values that have similar effects.

Consistency testing

Example

For example, in a communication network, two sources with an arrival rate of 100 packets per second each should cause approximately the same level of traffic in the network as four sources with arrival rate of 50 packets per second each.

If the model output shows a significant difference, either it should be possible to explain the difference from more detailed knowledge of the system, or the possibility of a modelling error should be investigated.

Model Validation

- Validation is the task of demonstrating that the model is a reasonable representation of the actual system: that it reproduces system behaviour with enough fidelity to satisfy analysis objectives.
- Whereas model verification techniques are general the approach taken to model validation is likely to be much more specific to the model/system and the question being asked.
- Model validation will be influenced by the objectives of the performance study (just like model development).

Level of detail

- A model is usually developed to analyse a particular problem and may therefore represent different parts of the system at different levels of abstraction.
- As a result, the model may have different levels of validity for different parts of the system across the full spectrum of system behaviour.

Aspects of model validation

For most models there are three separate aspects which should be considered during model validation:

- assumptions
- input parameter values and distributions
- output values and conclusions.

Aspects of model validation

For most models there are three separate aspects which should be considered during model validation:

- assumptions
- input parameter values and distributions
- output values and conclusions.

It may be difficult to achieve such a full validation of the model, especially if the system being modelled does not yet exist.

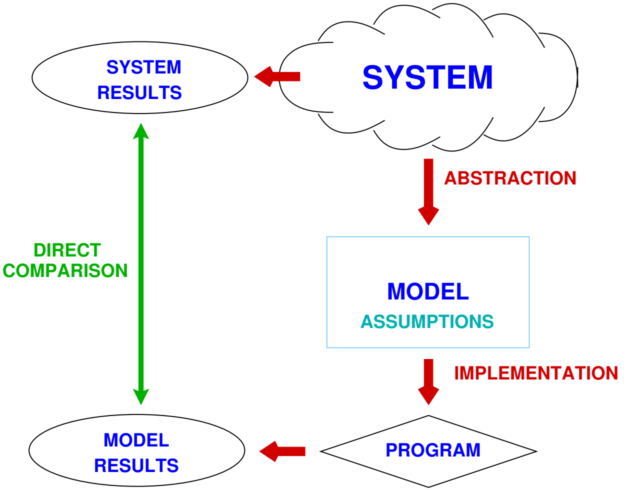
Aspects of model validation

For most models there are three separate aspects which should be considered during model validation:

- assumptions
- input parameter values and distributions
- output values and conclusions.

It may be difficult to achieve such a full validation of the model, especially if the system being modelled does not yet exist.

In practice if output values are available most validation exercises focus on this approach, at least initially.


SYSTEM
RESULTS
SYSTEM
ABSTRACTION
DIRECT
COMPARISON
MODEL
ASSUMPTIONS
IMPLEMENTATION
MODEL
RESULTS
PROGRAM

Approaches to model validation

Broadly speaking there are three approaches to model validation and any combination of them may be applied as appropriate to the different aspects of a particular model.

These approaches are:

- expert intuition
- real system measurements
- theoretical results/analysis.

In addition, ad hoc validation techniques may be established for a particular model and system.

Expert intuition

- Essentially using expert intuition to validate a model is similar to the use of one-step analysis during model verification.
- Here the examination of the model should ideally be led by someone other than the modeller, an "expert" with respect to the system, rather than with respect to the model.
- This might be the system designer, service engineers or marketing staff, depending on the stage of the system within its life-cycle.

Inspection and instrumentation

- Careful inspection of the model output, and model behaviour, will be assisted by one-step analysis, tracing and animation, in the case of simulation models, and the full steady state representation of the state space in the case of Markovian models.
- In either case, a model may be fully instrumented, meaning that every possible performance measure is extracted from the model for validation purposes regardless of the objectives of the performance study.

Real system measurements

- Comparison with a real system is the most reliable and preferred way to validate a simulation model.
- But it may be infeasible because the real system does not exist or because the measurements would be too expensive to carry out.
- Assumptions, input values, output values, workloads, configurations and system behaviour should all be compared with those observed in the real world.
- For simulation models, when full measurement data is available we may use trace-driven simulation to observe the model under exactly the same conditions as the real system.

Theoretical results/analysis

In the case of detailed Markovian models or simulation models it is sometimes possible to use a more abstract representation of the system to provide a crude validation of the model.

Theoretical results/analysis

In the case of detailed Markovian models or simulation models it is sometimes possible to use a more abstract representation of the system to provide a crude validation of the model.

In particular, if the results of an operational analysis, based on the operational laws coincide with model output it may be taken as evidence that the model behaves correctly.

Consistency between simulation and operational laws

- Another possible use for the operational laws is to check consistency within a set of results extracted from a simulation model.
- If a model is behaving correctly we would expect the measures extracted during the evolution of a model to obey the operational laws provided the usual assumptions hold.
- Failure of the operational laws would suggest that further investigation into the detailed behaviour of the model was necessary.

Consistency between simulation and operational laws

Example

The general residence time law can provide us with a simple validation of the model output values obtained for residence times at individual components if we know their respective visit counts, jobs behave homogeneously and we expect the model to be job flow balanced.

Consistency between analytical models and simulation

- At slightly more detail a simulation model may also be validated by comparing its output with a simple queueing network model of the same system.
- Conversely, (in academic work) Markovian models are often validated by comparing their outcome with that of a more detailed simulation model.

Caution

Validation of models against the results or behaviour of other models is a technique which should be used with care as both may be invalid in the sense that they both may not represent the behaviour of the real system accurately.

Invariants

- Another analytic approach is to determine invariants which must hold in every state of the system.
- For example, these invariants might capture a mutual exclusion condition or a conservation of work condition.
- Showing that the model always satisfies such an invariant is one way of increasing confidence in the model, and providing support for its validity.
- The disadvantage of such an approach is that it can be computationally expensive to carry out the necessary checks regularly within a model.

Model Validation Exercise

A performance analyst has constructed a simulation model of a communication network.

Preliminary experiments with the model have generated the performance estimates shown in the table below.

Throughput of System (msg/sec)	Utilisation of Node 1	Utilisation of Node 2	Trans. time Node 1 (ms/packet)	Trans. time Node2 (ms/packet)
25	50%	95%	2	3

Each message in the system generates 10 packets at Node 1 and 10 packets at Node 2.

Based on the data above do you think that the model is correct? Explain your reasoning.